**Project Report Template**

1.INTRODUCTION

* 1. Overview

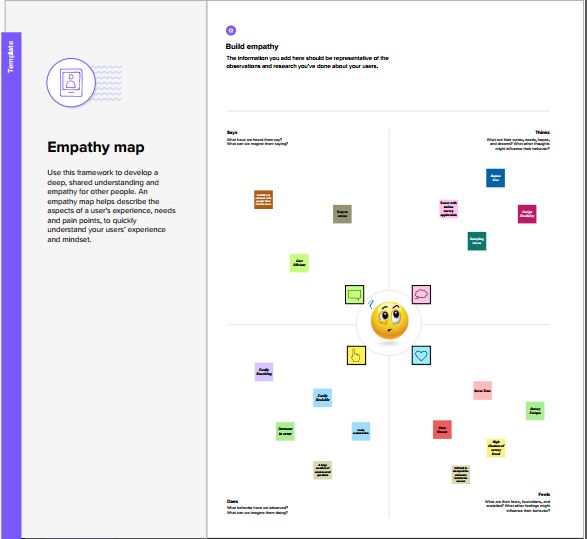
Material Design Components (MDC Android) offers designers and developers a way to implement Material Design in their Android application. Developed by a core team of engineers and UX designers at Google, these components enable a reliable development workflow to build beautiful and functional Android applications. Material design in Android is one of the key features that attracts and engages the customer towards the application.

1.2 Purpose

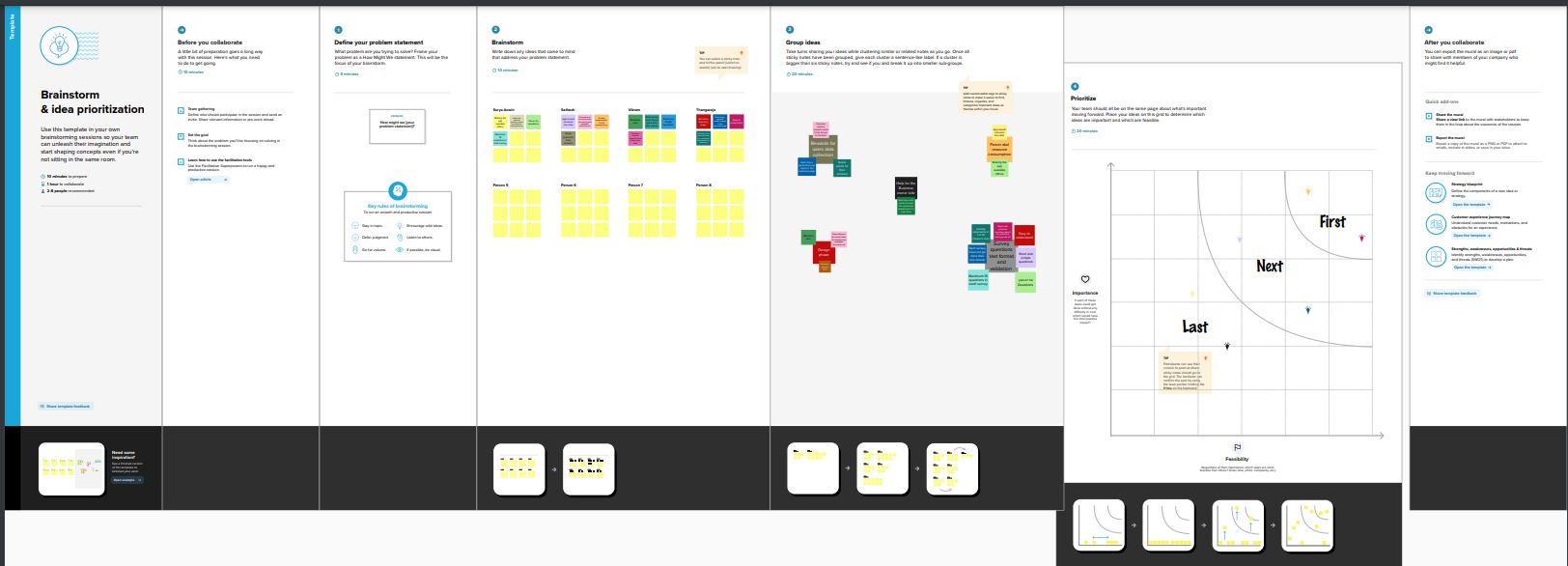
This is a special type of design, which is guided by Google. So in this article, it has been introduced to the basic things that need to be considered before designing or developing any Materialistic Android Application.

2.Problem Definition & Design Thinking

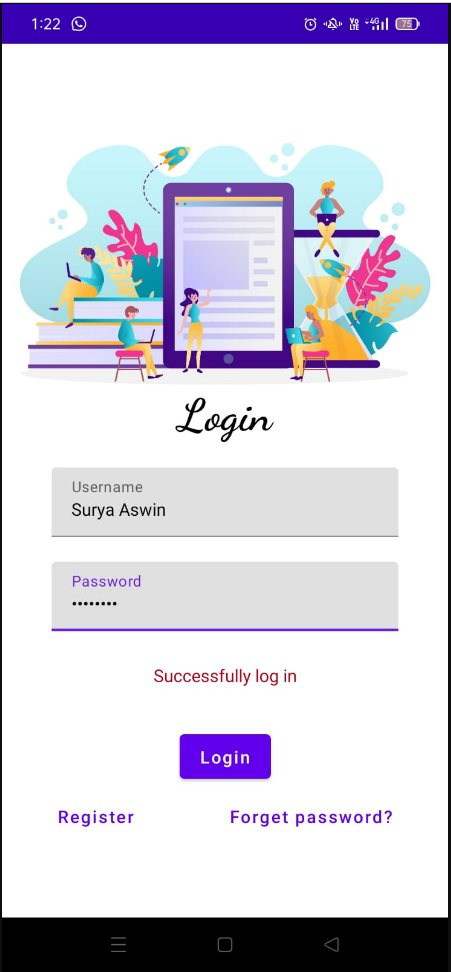
2.1 Empathy Map

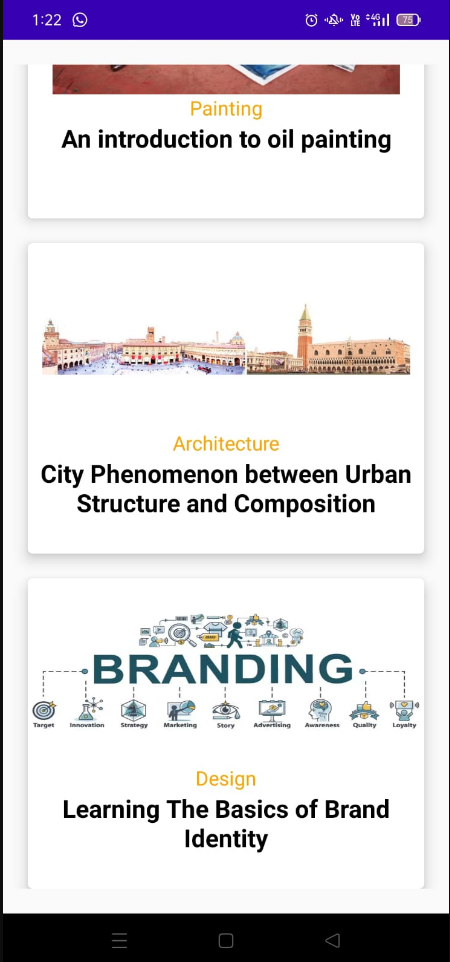


2.2 Ideation & Brainstorming Map



3.Result



4.ADVANTAGES & DISADVANTAGES

4.1 Advantages

As a set of best practices and as a design system, **Material Design offers a number of benefits**:

* It is a trustworthy and consistent package of detailed guidance for digital product UI designers.
* **Material Design has effectively solved most basic issues around usability, legibility and accessibility before you even begin**.
* With its level of detail, Material Design can enhance the understanding of the user interface, including its key components and behaviors – it can also be used to assess and analyze existing UIs.
* Having starting out as a system for Android apps, it is inherently mobile-focused.
* Material Design encourages designers and developers to make every digital product responsive.
* As a user-first approach to design, Material Design fits well with Agile frameworks of development, such as Scrum.
* Material Design is now sufficiently broad and flexible (and has such extensive documentation and support) that designers can produce best practice products without completely sacrificing their own creativity.

4.2 Disadvantages

* Material Design makes extensive use of animations, which can carry quite a lot of overhead
* This can make an interface lively, this style is prone to overuse and can be very distracting to users trying to get something done.
* For better or worse, Material Design is a Google-promoted framework and Android is a prominent early adopter, so the framework will probably never be dissociated from a perceptual relationship with Google.
* My intent here is not to criticize the framework unnecessarily...on the contrary, I really like and continue to use Material Design.
* It uses an unnecessarily high-contrast header background with multiple layers and off-grid lines which may be fun to look at once, but is functionally crappy because it makes the text hard to read and can get really annoying when the menu is opened multiple times
* Material design uses a lot of rectangles with sharp corners in its elements. This makes elements less visually appealing to users.

5. Application

* The Android operating system is a multi-user Linux system in which each app is a different user.
* By default, the system assigns each app a unique Linux user ID, which is used only by the system and is unknown to the app. The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.
* Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps.
* By default, every app runs in its own Linux process. The Android system starts the process when any of the app's components need to be executed, and then shuts down the process when it's no longer needed or when the system must recover memory for other apps.

7. Future Scope

We should expect to see an increase in note-taking gadgets in the classroom in various forms as note-taking technology grows more sophisticated with the addition of scribe pens and styli. We can see the possibility for transferring complicated material, such as handwritten text or drawings, digitally, with the rise of online, hybrid, and blended learning courses. Tools outside of the notetaking realm are also taking up note-taking features. For example, eBooks are now using notetaking features such as annotating, highlighting, and underlining to mimic the experience of reading a physical book.

6. Conclusion

From the research we understood that, currently, the popularization of the internet has led to a decline in revenue in the paper industry, but the pen and pencil industry’s revenue has grown in recent years. These conflicting numbers are inconclusive in regards to whether or not students are abandoning paper-basednotetaking. Many students are taking advantage of the electronics in the classroom. In conclusion, major benefits of the project are: No reach limitations, it provides a user-friendly GUI which results in retaining of users.

• Media attachment support and inbuilt scanner to scan notes

• Flexibility for users to generate various files of notes such as .png, .pdf, .txt

• Inbuilt Passcode and biometric security.

• Lightweight SQLite database.

8.Appendix

Source code:-

Code

Email data base helper.kt

package com.example.emailapplication

import android.annotation.SuppressLint

import android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper

class EmailDatabaseHelper(context: Context) :

SQLiteOpenHelper(context, DATABASE\_NAME, null,DATABASE\_VERSION){

companion object {

private const val DATABASE\_VERSION = 1

private const val DATABASE\_NAME = "EmailDatabase.db"

private const val TABLE\_NAME = "email\_table"

private const val COLUMN\_ID = "id"

private const val COLUMN\_RECEIVER\_MAIL = "receiver\_mail"

private const val COLUMN\_SUBJECT = "subject"

private const val COLUMN\_BODY = "body"

}

override fun onCreate(db: SQLiteDatabase?) {

val createTable = "CREATE TABLE $TABLE\_NAME (" +

"${COLUMN\_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " +

"${COLUMN\_RECEIVER\_MAIL} Text, " +

"${COLUMN\_SUBJECT} TEXT ," +

"${COLUMN\_BODY} TEXT " +

")"

db?.execSQL(createTable)

}

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME")

onCreate(db)

}

fun insertEmail(email: Email) {

val db = writableDatabase

val values = ContentValues()

values.put(COLUMN\_RECEIVER\_MAIL, email.recevierMail)

values.put(COLUMN\_SUBJECT, email.subject)

values.put(COLUMN\_BODY, email.body)

db.insert(TABLE\_NAME, null, values)

db.close()

}

@SuppressLint("Range")

fun getEmailBySubject(subject: String): Email? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_SUBJECT = ?", arrayOf(subject))

var email: Email? = null

if (cursor.moveToFirst()) {

email = Email(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

recevierMail = cursor.getString(cursor.getColumnIndex(COLUMN\_RECEIVER\_MAIL)),

subject = cursor.getString(cursor.getColumnIndex(COLUMN\_SUBJECT)),

body = cursor.getString(cursor.getColumnIndex(COLUMN\_BODY)),

)

}

cursor.close()

db.close()

return email

}

@SuppressLint("Range")

fun getEmailById(id: Int): Email? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString()))

var email: Email? = null

if (cursor.moveToFirst()) {

email = Email(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

recevierMail = cursor.getString(cursor.getColumnIndex(COLUMN\_RECEIVER\_MAIL)),

subject = cursor.getString(cursor.getColumnIndex(COLUMN\_SUBJECT)),

body = cursor.getString(cursor.getColumnIndex(COLUMN\_BODY)),

)

}

cursor.close()

db.close()

return email

}

@SuppressLint("Range")

fun getAllEmails(): List<Email> {

val emails = mutableListOf<Email>()

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null)

if (cursor.moveToFirst()) {

do {

val email = Email(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

recevierMail = cursor.getString(cursor.getColumnIndex(COLUMN\_RECEIVER\_MAIL)),

subject = cursor.getString(cursor.getColumnIndex(COLUMN\_SUBJECT)),

body = cursor.getString(cursor.getColumnIndex(COLUMN\_BODY)),

)

emails.add(email)

} while (cursor.moveToNext())

}

cursor.close()

db.close()

return emails

}

}

Email.kt

package com.example.emailapplication

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "email\_table")

data class Email(

@PrimaryKey(autoGenerate = true) val id: Int?,

@ColumnInfo(name = "receiver\_mail") val recevierMail: String?,

@ColumnInfo(name = "subject") val subject: String?,

@ColumnInfo(name = "body") val body: String?,

)

Login activity.kt

package com.example.emailapplication

import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.background

import androidx.compose.foundation.layout.\*

import androidx.compose.material.\*

import androidx.compose.runtime.\*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontFamily

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.input.PasswordVisualTransformation

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

import com.example.emailapplication.ui.theme.EmailApplicationTheme

class LoginActivity : ComponentActivity() {

private lateinit var databaseHelper: UserDatabaseHelper

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

databaseHelper = UserDatabaseHelper(this)

setContent {

LoginScreen(this, databaseHelper)

}

}

}

@Composable

fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {

var username by remember { mutableStateOf("") }

var password by remember { mutableStateOf("") }

var error by remember { mutableStateOf("") }

Column(

modifier = Modifier.fillMaxSize().background(Color.White),

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.Center

) {

Image(

painterResource(id = R.drawable.email\_login), contentDescription = ""

)

Text(

fontSize = 36.sp,

fontWeight = FontWeight.ExtraBold,

fontFamily = FontFamily.Cursive,

text = "Login"

)

Spacer(modifier = Modifier.height(10.dp))

TextField(

value = username,

onValueChange = { username = it },

label = { Text("Username") },

modifier = Modifier.padding(10.dp)

.width(280.dp)

)

TextField(

value = password,

onValueChange = { password = it },

label = { Text("Password") },

visualTransformation = PasswordVisualTransformation(),

modifier = Modifier.padding(10.dp)

.width(280.dp)

)

if (error.isNotEmpty()) {

Text(

text = error,

color = MaterialTheme.colors.error,

modifier = Modifier.padding(vertical = 16.dp)

)

}

Button(

onClick = {

if (username.isNotEmpty() && password.isNotEmpty()) {

val user = databaseHelper.getUserByUsername(username)

if (user != null && user.password == password) {

error = "Successfully log in"

context.startActivity(

Intent(

context,

MainActivity::class.java

)

)

//onLoginSuccess()

}

} else {

error = "Please fill all fields"

}

},

colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef)),

modifier = Modifier.padding(top = 16.dp)

) {

Text(text = "Login")

}

Row {

TextButton(onClick = {context.startActivity(

Intent(

context,

RegisterActivity::class.java

)

)}

)

{ Text(color = Color(0xFF31539a),text = "Sign up") }

TextButton(onClick = {

})

{

Spacer(modifier = Modifier.width(60.dp))

Text(color = Color(0xFF31539a),text = "Forget password?")

}

}

}

}

private fun startMainPage(context: Context) {

val intent = Intent(context, MainActivity::class.java)

ContextCompat.startActivity(context, intent, null)

}

EmailDao.kt

package com.example.emailapplication

import androidx.room.\*

@Dao

interface EmailDao {

@Query("SELECT \* FROM email\_table WHERE subject= :subject")

suspend fun getOrderBySubject(subject: String): Email?

@Insert(onConflict = OnConflictStrategy.REPLACE)

suspend fun insertEmail(email: Email)

@Update

suspend fun updateEmail(email: Email)

@Delete

suspend fun deleteEmail(email: Email)

}

Main Activity.kt

package com.example.emailapplication

import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.background

import androidx.compose.foundation.layout.\*

import androidx.compose.material.\*

import androidx.compose.runtime.Composable

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

import androidx.core.content.ContextCompat.startActivity

import com.example.emailapplication.ui.theme.EmailApplicationTheme

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContent {

// A surface container using the 'background' color from the theme

Surface(

modifier = Modifier.fillMaxSize().background(Color.White),

) {

Email(this)

}

}

}

}

@Composable

fun Email(context: Context) {

Text(

text = "Home Screen",

modifier = Modifier.padding(top = 74.dp, start = 100.dp, bottom = 24.dp),

color = Color.Black,

fontWeight = FontWeight.Bold,

fontSize = 32.sp

)

Column(

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.Center

) {

Image(

painterResource(id = R.drawable.home\_screen), contentDescription = ""

)

Button(onClick = {

context.startActivity(

Intent(

context,

SendMailActivity::class.java

)

)

},

colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFadbef4))

) {

Text(

text = "Send Email",

modifier = Modifier.padding(10.dp),

color = Color.Black,

fontSize = 15.sp

)

}

Spacer(modifier = Modifier.height(20.dp))

Button(onClick = {

context.startActivity(

Intent(

context,

ViewMailActivity::class.java

)

)

},

colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFadbef4))

) {

Text(

text = "View Emails",

modifier = Modifier.padding(10.dp),

color = Color.Black,

fontSize = 15.sp

)

}

}

}

RegisterActivity.kt

package com.example.emailapplication

import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.background

import androidx.compose.foundation.layout.\*

import androidx.compose.material.\*

import androidx.compose.runtime.\*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontFamily

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.input.PasswordVisualTransformation

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

import com.example.emailapplication.ui.theme.EmailApplicationTheme

class RegisterActivity : ComponentActivity() {

private lateinit var databaseHelper: UserDatabaseHelper

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

databaseHelper = UserDatabaseHelper(this)

setContent {

RegistrationScreen(this, databaseHelper)

}

}

}

@Composable

fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) {

var username by remember { mutableStateOf("") }

var password by remember { mutableStateOf("") }

var email by remember { mutableStateOf("") }

var error by remember { mutableStateOf("") }

Column(

modifier = Modifier.fillMaxSize().background(Color.White),

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.Center

) {

Image(

painterResource(id = R.drawable.email\_signup), contentDescription = "",

modifier = Modifier.height(300.dp)

)

Text(

fontSize = 36.sp,

fontWeight = FontWeight.ExtraBold,

fontFamily = FontFamily.Cursive,

text = "Register"

)

Spacer(modifier = Modifier.height(10.dp))

TextField(

value = username,

onValueChange = { username = it },

label = { Text("Username") },

modifier = Modifier

.padding(10.dp)

.width(280.dp)

)

TextField(

value = email,

onValueChange = { email = it },

label = { Text("Email") },

modifier = Modifier

.padding(10.dp)

.width(280.dp)

)

TextField(

value = password,

onValueChange = { password = it },

label = { Text("Password") },

visualTransformation = PasswordVisualTransformation(),

modifier = Modifier

.padding(10.dp)

.width(280.dp)

)

if (error.isNotEmpty()) {

Text(

text = error,

color = MaterialTheme.colors.error,

modifier = Modifier.padding(vertical = 16.dp)

)

}

Button(

onClick = {

if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {

val user = User(

id = null,

firstName = username,

lastName = null,

email = email,

password = password

)

databaseHelper.insertUser(user)

error = "User registered successfully"

// Start LoginActivity using the current context

context.startActivity(

Intent(

context,

LoginActivity::class.java

)

)

} else {

error = "Please fill all fields"

}

},

colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef)),

modifier = Modifier.padding(top = 16.dp)

) {

Text(text = "Register")

}

Spacer(modifier = Modifier.width(10.dp))

Spacer(modifier = Modifier.height(10.dp))

Row() {

Text(

modifier = Modifier.padding(top = 14.dp), text = "Have an account?"

)

TextButton(onClick = {

context.startActivity(

Intent(

context,

LoginActivity::class.java

)

)

})

{

Spacer(modifier = Modifier.width(10.dp))

Text(color = Color(0xFF31539a),text = "Log in")

}

}

}

}

private fun startLoginActivity(context: Context) {

val intent = Intent(context, LoginActivity::class.java)

ContextCompat.startActivity(context, intent, null)

}

SendMailActivity.kt

package com.example.emailapplication

import android.annotation.SuppressLint

import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.layout.\*

import androidx.compose.material.\*

import androidx.compose.runtime.\*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.platform.LocalContext

import androidx.compose.ui.text.TextStyle

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import com.example.emailapplication.ui.theme.EmailApplicationTheme

class SendMailActivity : ComponentActivity() {

private lateinit var databaseHelper: EmailDatabaseHelper

@SuppressLint("UnusedMaterialScaffoldPaddingParameter")

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

databaseHelper = EmailDatabaseHelper(this)

setContent {

Scaffold(

// in scaffold we are specifying top bar.

topBar = {

// inside top bar we are specifying

// background color.

TopAppBar(backgroundColor = Color(0xFFadbef4), modifier = Modifier.height(80.dp),

// along with that we are specifying

// title for our top bar.

title = {

// in the top bar we are specifying

// title as a text

Text(

// on below line we are specifying

// text to display in top app bar.

text = "Send Mail",

fontSize = 32.sp,

color = Color.Black,

// on below line we are specifying

// modifier to fill max width.

modifier = Modifier.fillMaxWidth(),

// on below line we are

// specifying text alignment.

textAlign = TextAlign.Center,

)

}

)

}

) {

// on below line we are

// calling method to display UI.

openEmailer(this,databaseHelper)

}

}

}

}

@Composable

fun openEmailer(context: Context, databaseHelper: EmailDatabaseHelper) {

// in the below line, we are

// creating variables for URL

var recevierMail by remember {mutableStateOf("") }

var subject by remember {mutableStateOf("") }

var body by remember {mutableStateOf("") }

var error by remember { mutableStateOf("") }

// on below line we are creating

// a variable for a context

val ctx = LocalContext.current

// on below line we are creating a column

Column(

// on below line we are specifying modifier

// and setting max height and max width

// for our column

modifier = Modifier

.fillMaxSize()

.padding(top = 55.dp, bottom = 25.dp, start = 25.dp, end = 25.dp),

horizontalAlignment = Alignment.Start

) {

// on the below line, we are

// creating a text field.

Text(text = "Receiver Email-Id",

fontWeight = FontWeight.Bold,

fontSize = 16.sp)

TextField(

// on below line we are specifying

// value for our text field.

value = recevierMail,

// on below line we are adding on value

// change for text field.

onValueChange = { recevierMail = it },

// on below line we are adding place holder as text

label = { Text(text = "Email address") },

placeholder = { Text(text = "abc@gmail.com") },

// on below line we are adding modifier to it

// and adding padding to it and filling max width

modifier = Modifier

.padding(16.dp)

.fillMaxWidth(),

// on below line we are adding text style

// specifying color and font size to it.

textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),

// on below line we are

// adding single line to it.

singleLine = true,

)

// on below line adding a spacer.

Spacer(modifier = Modifier.height(10.dp))

Text(text = "Mail Subject",

fontWeight = FontWeight.Bold,

fontSize = 16.sp)

// on the below line, we are creating a text field.

TextField(

// on below line we are specifying

// value for our text field.

value = subject,

// on below line we are adding on value change

// for text field.

onValueChange = { subject = it },

// on below line we are adding place holder as text

placeholder = { Text(text = "Subject") },

// on below line we are adding modifier to it

// and adding padding to it and filling max width

modifier = Modifier

.padding(16.dp)

.fillMaxWidth(),

// on below line we are adding text style

// specifying color and font size to it.

textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),

// on below line we are

// adding single line to it.

singleLine = true,

)

// on below line adding a spacer.

Spacer(modifier = Modifier.height(10.dp))

Text(text = "Mail Body",

fontWeight = FontWeight.Bold,

fontSize = 16.sp)

// on the below line, we are creating a text field.

TextField(

// on below line we are specifying

// value for our text field.

value = body,

// on below line we are adding on value

// change for text field.

onValueChange = { body = it },

// on below line we are adding place holder as text

placeholder = { Text(text = "Body") },

// on below line we are adding modifier to it

// and adding padding to it and filling max width

modifier = Modifier

.padding(16.dp)

.fillMaxWidth(),

// on below line we are adding text style

// specifying color and font size to it.

textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),

// on below line we are

// adding single line to it.

singleLine = true,

)

// on below line adding a spacer.

Spacer(modifier = Modifier.height(20.dp))

// on below line adding a

// button to send an email

Button(onClick = {

if( recevierMail.isNotEmpty() && subject.isNotEmpty() && body.isNotEmpty()) {

val email = Email(

id = null,

recevierMail = recevierMail,

subject = subject,

body = body

)

databaseHelper.insertEmail(email)

error = "Mail Saved"

} else {

error = "Please fill all fields"

}

// on below line we are creating

// an intent to send an email

val i = Intent(Intent.ACTION\_SEND)

// on below line we are passing email address,

// email subject and email body

val emailAddress = arrayOf(recevierMail)

i.putExtra(Intent.EXTRA\_EMAIL,emailAddress)

i.putExtra(Intent.EXTRA\_SUBJECT,subject)

i.putExtra(Intent.EXTRA\_TEXT,body)

// on below line we are

// setting type of intent

i.setType("message/rfc822")

// on the below line we are starting our activity to open email application.

ctx.startActivity(Intent.createChooser(i,"Choose an Email client : "))

},

colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef))

) {

// on the below line creating a text for our button.

Text(

// on below line adding a text ,

// padding, color and font size.

text = "Send Email",

modifier = Modifier.padding(10.dp),

color = Color.Black,

fontSize = 15.sp

)

}

}

}

UserDao.kt

package com.example.emailapplication

import androidx.room.\*

@Dao

interface UserDao {

@Query("SELECT \* FROM user\_table WHERE email = :email")

suspend fun getUserByEmail(email: String): User?

@Insert(onConflict = OnConflictStrategy.REPLACE)

suspend fun insertUser(user: User)

@Update

suspend fun updateUser(user: User)

@Delete

suspend fun deleteUser(user: User)

}

User.kt

package com.example.emailapplication

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "user\_table")

data class User(

@PrimaryKey(autoGenerate = true) val id: Int?,

@ColumnInfo(name = "first\_name") val firstName: String?,

@ColumnInfo(name = "last\_name") val lastName: String?,

@ColumnInfo(name = "email") val email: String?,

@ColumnInfo(name = "password") val password: String?,

)

UserDataBase.kt

package com.example.emailapplication

import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)

abstract class UserDatabase : RoomDatabase() {

abstract fun userDao(): UserDao

companion object {

@Volatile

private var instance: UserDatabase? = null

fun getDatabase(context: Context): UserDatabase {

return instance ?: synchronized(this) {

val newInstance = Room.databaseBuilder(

context.applicationContext,

UserDatabase::class.java,

"user\_database"

).build()

instance = newInstance

newInstance

}

}

}

}

UserDatabaseHelper.kt

package com.example.emailapplication

import android.annotation.SuppressLint

import android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :

SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {

companion object {

private const val DATABASE\_VERSION = 1

private const val DATABASE\_NAME = "UserDatabase.db"

private const val TABLE\_NAME = "user\_table"

private const val COLUMN\_ID = "id"

private const val COLUMN\_FIRST\_NAME = "first\_name"

private const val COLUMN\_LAST\_NAME = "last\_name"

private const val COLUMN\_EMAIL = "email"

private const val COLUMN\_PASSWORD = "password"

}

override fun onCreate(db: SQLiteDatabase?) {

val createTable = "CREATE TABLE $TABLE\_NAME (" +

"$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +

"$COLUMN\_FIRST\_NAME TEXT, " +

"$COLUMN\_LAST\_NAME TEXT, " +

"$COLUMN\_EMAIL TEXT, " +

"$COLUMN\_PASSWORD TEXT" +

")"

db?.execSQL(createTable)

}

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME")

onCreate(db)

}

fun insertUser(user: User) {

val db = writableDatabase

val values = ContentValues()

values.put(COLUMN\_FIRST\_NAME, user.firstName)

values.put(COLUMN\_LAST\_NAME, user.lastName)

values.put(COLUMN\_EMAIL, user.email)

values.put(COLUMN\_PASSWORD, user.password)

db.insert(TABLE\_NAME, null, values)

db.close()

}

@SuppressLint("Range")

fun getUserByUsername(username: String): User? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_FIRST\_NAME = ?", arrayOf(username))

var user: User? = null

if (cursor.moveToFirst()) {

user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

}

cursor.close()

db.close()

return user

}

@SuppressLint("Range")

fun getUserById(id: Int): User? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString()))

var user: User? = null

if (cursor.moveToFirst()) {

user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

}

cursor.close()

db.close()

return user

}

@SuppressLint("Range")

fun getAllUsers(): List<User> {

val users = mutableListOf<User>()

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null)

if (cursor.moveToFirst()) {

do {

val user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

users.add(user)

} while (cursor.moveToNext())

}

cursor.close()

db.close()

return users

}

}

ViewMailActivity.kt

package com.example.emailapplication

import android.annotation.SuppressLint

import android.os.Bundle

import android.util.Log

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.\*

import androidx.compose.foundation.layout.R

import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.LazyRow

import androidx.compose.foundation.lazy.items

import androidx.compose.material.\*

import androidx.compose.runtime.Composable

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import com.example.emailapplication.ui.theme.EmailApplicationTheme

class ViewMailActivity : ComponentActivity() {

private lateinit var emailDatabaseHelper: EmailDatabaseHelper

@SuppressLint("UnusedMaterialScaffoldPaddingParameter")

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

emailDatabaseHelper = EmailDatabaseHelper(this)

setContent {

Scaffold(

// in scaffold we are specifying top bar.

topBar = {

// inside top bar we are specifying

// background color.

TopAppBar(backgroundColor = Color(0xFFadbef4), modifier = Modifier.height(80.dp),

// along with that we are specifying

// title for our top bar.

title = {

// in the top bar we are specifying

// title as a text

Text(

// on below line we are specifying

// text to display in top app bar.

text = "View Mails",

fontSize = 32.sp,

color = Color.Black,

// on below line we are specifying

// modifier to fill max width.

modifier = Modifier.fillMaxWidth(),

// on below line we are

// specifying text alignment.

textAlign = TextAlign.Center,

)

}

)

}

) {

val data = emailDatabaseHelper.getAllEmails();

Log.d("swathi", data.toString())

val email = emailDatabaseHelper.getAllEmails()

ListListScopeSample(email)

}

}

}

}

@Composable

fun ListListScopeSample(email: List<Email>) {

LazyRow(

modifier = Modifier

.fillMaxSize(),

horizontalArrangement = Arrangement.SpaceBetween

) {

item {

LazyColumn {

items(email) { email ->

Column(

modifier = Modifier.padding(

top = 16.dp,

start = 48.dp,

bottom = 20.dp

)

) {

Text("Receiver\_Mail: ${email.recevierMail}", fontWeight = FontWeight.Bold)

Text("Subject: ${email.subject}")

Text("Body: ${email.body}")

}

}

}

}

}

}

Build.gradile

plugins {

id 'com.android.application'

id 'org.jetbrains.kotlin.android'

}

android {

namespace 'com.example.emailapplication'

compileSdk 33

defaultConfig {

applicationId "com.example.emailapplication"

minSdk 21

targetSdk 33

versionCode 1

versionName "1.0"

testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

vectorDrawables {

useSupportLibrary true

}

}

buildTypes {

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

}

}

compileOptions {

sourceCompatibility JavaVersion.VERSION\_1\_8

targetCompatibility JavaVersion.VERSION\_1\_8

}

kotlinOptions {

jvmTarget = '1.8'

}

buildFeatures {

compose true

}

composeOptions {

kotlinCompilerExtensionVersion '1.2.0'

}

packagingOptions {

resources {

excludes += '/META-INF/{AL2.0,LGPL2.1}'

}

}

}

dependencies {

implementation 'androidx.core:core-ktx:1.7.0'

implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'

implementation 'androidx.activity:activity-compose:1.3.1'

implementation "androidx.compose.ui:ui:$compose\_ui\_version"

implementation "androidx.compose.ui:ui-tooling-preview:$compose\_ui\_version"

implementation 'androidx.compose.material:material:1.2.0'

implementation 'androidx.room:room-common:2.5.0'

implementation 'androidx.room:room-ktx:2.5.0'

testImplementation 'junit:junit:4.13.2'

androidTestImplementation 'androidx.test.ext:junit:1.1.5'

androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'

androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose\_ui\_version"

debugImplementation "androidx.compose.ui:ui-tooling:$compose\_ui\_version"

debugImplementation "androidx.compose.ui:ui-test-manifest:$compose\_ui\_version"

}